Survey of Data Mining Approaches to User Modeling for Adaptive Hypermedia

Enrique Frias-Martinez, Sherry Y. Chen, and Xiaohui Liu

***Abstract*—The ability of an adaptive hypermedia system to create tailored environments depends mainly on the amount and accuracy of information stored in each user model. Some of the difficulties that user modeling faces are the amount of data available to create user models, the adequacy of the data, the noise within that data, and the necessity of capturing the imprecise nature of human behavior. Data mining and machine learning techniques have the ability to handle large amounts of data and to process uncertainty. These characteristics make these techniques suitable for automatic generation of user models that simulate human decision making. This paper surveys different data mining techniques that can be used to efficiently and accurately capture user behavior. The paper also presents guidelines that show which techniques may be used more efficiently according to the task implemented by the application.**

***Index Terms*—Adaptive hypermedia (AH), data mining, machine learning, user modeling (UM).**

I. Introduction

ADAPTIVE hypermedia (AH) can be defined as the technology that allows to personalize for each individual user of a hypermedia application the content and presentation of the application according to user preferences and characteristics [68]–[70].

The process of personalization is defined as the way in which information and services can be tailored to match the unique and specific needs of an individual or a community [17]. Personalization is about building customer loyalty by developing a meaningful one-to-one relationship, by understanding the needs of each individual and helping satisfy a goal that efficiently and knowledgeably addresses each individual's need in a given context [75]. The more information a user model has, the better the content, and presentation will be personalized. A user model is created through a user modeling (UM) process in which unobservable information about a user is inferred from observable information from that user, e.g., using the interactions with the system [97], [98]. User models can be created using a user-guided approach, in which the models are directly created using the information provided by each user, or an automatic approach, in which the process of creating a user model is controlled by the system and is hidden from the user. The user-guided approach produces adaptable services and adaptable user models [32], while the automatic approach produces adaptive services and adaptive user models [16], [32]. In general, a user model will contain some adaptive and some adaptable elements. Ideally, the set of adaptable elements should be reduced to the minimum possible (elements like age, gender, favorite background color, etc.), while other elements (favorite topics, patterns of behavior, etc.) should be created by a learning process. These concepts have also been presented in the literature as implicit and explicit UM acquisition [73].

The problem of UM can be implemented using an automatic approach because a typical user exhibits patterns when accessing a hypermedia system and the set of interactions containing those patterns can be stored in a log database in the server. In this context, machine learning and data mining techniques can be applied to recognize regularities in user trails and to integrate them as part of the user model. Machine learning encompasses techniques where a machine acquires/learns knowledge from its previous experience [93]. The output of a machine learning technique is a structural description of what has been learned that can be used to explain the original data and to make predictions. From this perspective, data mining and other machine learning techniques make it possible to automatically create user models for the implementation of AH services. These techniques make it possible to create user models in an environment, such as a hypermedia application, in which, usually, users are not willing to give feedback of their actions.

This fact leads researchers to consider the different techniques that can be used to capture and model user behavior and which elements of the behavior of a user each one of those techniques can capture. In general, the adaptive service that is going to be implemented determines the information that a user model should contain. Once that is known, and taking into account the characteristics of the available data, different data mining/machine learning techniques can be chosen to capture the necessary patterns.

This paper has surveyed the development of user models using data mining and machine learning techniques from 1999 to 2004, focusing on the main journals and conferences for UM, mainly:

- User Modeling and User-Adapted Interaction;
- International Conference on User Modeling, IEEE Transactions on Neural Networks;
- Workshop of Intelligent Techniques for Web Personalization (part of IJCAI International Joint Conference of Artificial Intelligence); and
- International Workshop on Knowledge Discovery on the WEB (WEBKDD, part of the ACM SIGKDD

Manuscript received October 1, 2004; revised February 28, 2005 and July 8, 2005. This work was supported in part by the U.K. Arts and Humanities Research Board under Grant MRG/AN9183/APN16300. This paper was recommended by Associate Editor M. Last.

The authors are with the Department of Information Systems and Computing, Brunel University, Uxbridge, UB8 3PH, U.K. (e-mail: enrique.frias-martinez@brunel.ac.uk; sherry.chen@brunel.ac.uk; xiaohui.liu@brunel.ac.uk).

Digital Object Identifier 10.1109/TSMCC.2006.879391

International Conference on Knowledge Discovery and Data Mining).

The paper's intentions are: 1) to give an up-to-date view of data mining techniques applied to UM and highlight their potential advantages and limitations and 2) to give basic guidelines about which techniques can be useful for a given adaptive application. The paper complements the results of [97], which reviewed the use of predictive statistical models for UM.

The organization of the paper is as follows. The paper first defines the concept of user model, its relevance for AH, and the basic steps for the automatic creation of user models. After that a set of unsupervised and supervised techniques are presented. For each technique, we present its theoretical background, its pros and cons, and its applications in the field of UM. Next, we develop a guideline on how to create a user model according to the needs of the AH application that is going to be implemented. Finally, conclusions are drawn at the end of the paper.

II. UM

A user model should capture the behavior (patterns, goals, interesting topics, etc.) a user shows when interacting with the Web. A user model is defined as a set of information structures designed to represent one or more of the following elements [49]: 1) representation of goals, plans, preferences, tasks, and/or abilities about one or more types of users; 2) representation of relevant common characteristics of users pertaining to specific user subgroups or stereotypes; 3) the classification of a user in one or more of these subgroups or stereotypes; 4) the recording of user behavior; 5) the formation of assumptions about the user based on the interaction history; and/or 6) the generalization of the interaction histories of many users into groups.

A. UM and AH

AH allows to personalize to each individual the content and presentation of a Web site. The architecture of an AH system is usually divided into two parts: the server side and the client side. The server side generates the user models from a database containing the interactions of the users with the system and the personal data/preferences that each user has given to the system. Also, user models can contain knowledge introduced by the designer (in the form of rules, for example). These user models, in combination with a hypermedia database, are used by the "Decision Making and Personalization Engine" module to identify user needs, decide on the types of adaptation to be performed, and communicate them to an adaptive interface. Fig. 1 presents the architecture of a generic AH system.

A personalized hypermedia system, by its very nature, should respond in real time to user inputs. To do so, the architecture of the system should provide a quick access to the right information at the right time.

AH uses the knowledge given by user models to implement one or more of the two basic types of adaptive tasks.

1) *Recommendation (R)*. Recommendation is the capability of suggesting interesting elements to a user based on some information, e.g., from the items to be recommended or from the behavior of other users. Recommendation is also known in the literature as collaborative filtering.
2) *Classification (C)*. Classification builds a model that maps or classifies data items into one of several predefined classes. Classification is done using only data related to that particular item. This knowledge can be used to tailor the services of each user stereotype (class). In the literature, classification has also been presented as content-based filtering.

Both recommendation and classification are types of filtering applications.

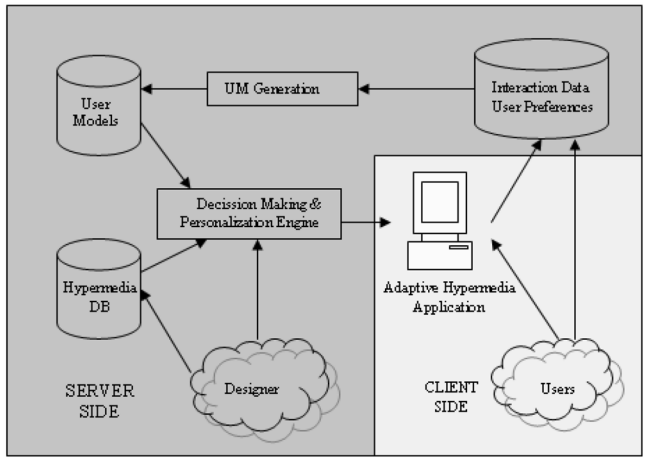

Fig. 1. Generic architecture of an AH application.

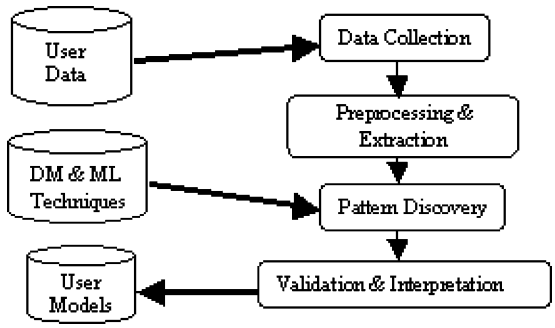

Fig. 2. Steps for automatic generation of user models.

B. Automatic Generation of User Models

One of the processes presented in Fig. 1 is the automatic generation of user models from the interaction data between the users and the system (done by the UM Generation module). Fig. 2 presents the basic steps of this module: 1) data collection; 2) preprocessing; 3) pattern discovery; and 4) validation and interpretation. A brief discussion of each of these follows.

1) *Data collection*. In this stage, user data are gathered. For automatic user modeling, the data collected includes: data regarding the interaction between the user and the Web, data regarding the environment of the user when interacting with the Web, direct feedback given by the user, etc.
2) *Data preprocessing/information extraction*. The information obtained in the previous stage cannot be

directly processed. It needs to be cleaned from noise and inconsistencies in order to be used as the input of the next phase. For user modeling, this involves mainly user identification and session reconstruction. This stage is aimed at obtaining, from the data available, the semantic content about the user interaction with the system. Also, in this phase, the data extracted should be adapted to the data structure used by standard pattern discovery algorithms used in the next step.

3) *Pattern discovery*. In this phase, machine learning and data mining techniques are applied to the data obtained in the previous stage in order to capture user behavior. The output of this stage is a set of structural descriptions of what have been learned about user behavior and user interests. These descriptions constitute the base of a user model. Different techniques will capture different user properties and will express them in different ways. The knowledge needed to implement an adaptive service will determine, among other factors, which techniques to apply in this phase.
4) *Validation and interpretation*. In this phase, the structures obtained in the pattern discovery stage are analyzed and interpreted. The patterns discovered can be interpreted and validated, using domain knowledge and visualization tools, in order to test the importance and usability of the knowledge obtained. In general, this process is done with the help of a UM designer.

User model heuristics are used in each step of the process to extract, adapt, and present the knowledge in a relevant way. The process of generation of user models using data mining/machine learning techniques can be seen as a standard process of extracting knowledge from data where UM is used as a wrapper for the entire process.

C. Data Mining and Its Relevance to UM

As presented in Fig. 2, the phase of pattern discovery finds out relevant information about the behavior of a user (or set of users) when interacting with the Web. Data mining and machine learning techniques are ideal for that process because they are designed to represent what have been learned from the input data with a structural representation. This representation stores the knowledge needed to implement the two types of tasks previously described.

Although the application of machine learning and data mining techniques works really well for modeling user behavior, it also faces some problems [91]. Among those challenges is the problem of needing large data sets, the problem of labeling data for supervised machine leaning techniques, and the problem of computational complexity.

Each data mining/machine learning technique will capture different relationships among the data available and will express the results using different data structures. The key question is to find out which patterns need to be captured in order to implement an adaptive service. It is important, in order to choose a suitable learning method, to know what knowledge is captured by each technique and how that knowledge can be used to implement the two basic tasks. Also, the choice of learning method depends largely on the type of training data available. The main distinction in machine learning research is between supervised and unsupervised learning.

Supervised learning requires the training data to be preclassified. This means that each training item is assigned a unique label, signifying the class to which the item belongs. Given these data, the learning algorithm builds a characteristic description for each class, covering the examples of this class. The important feature of this approach is that the class descriptions are built conditional to the preclassification of the examples in the training set.

In contrast, unsupervised learning methods do not require preclassification of the training examples. These methods form clusters of examples, which share common characteristics. The main difference to supervised learning is that categories are not known in advance, but constructed by the learner. When the cohesion of a cluster is high, i.e., the examples in it are similar, it defines a new class.

The rest of the paper presents how data mining and machine learning techniques have been used for UM: which knowledge can be captured with each technique, examples of applications, and its limits and strengths. The techniques presented are divided into two groups: unsupervised, which includes clustering (hierarchical, nonhierarchical, and fuzzy clustering) and association rules, and supervised, which includes decision trees/classification rules, k-nearest neighbor (k-NN), neural networks and support vector machines (SVM).

III. Unsupervised Approaches to UM

The main unsupervised techniques are clustering and association rules. Clustering comprises a wide variety of different techniques based on the same concept. A collection of different clustering techniques and its variations can be found in [42] and [43].

A. Clustering for UM

The task of clustering is to structure a given set of unclassified instances (data vectors) by creating concepts, based on similarities found on the training data.

A clustering algorithm finds the set of concepts that cover all examples verifying that: 1) the similarity between examples of the same concepts is maximized; and 2) the similarity between examples of different concepts is minimized. In a cluster algorithm, the key element is how to obtain the similarity between two items of the training set.

Clustering techniques can be classified into hard clustering and fuzzy clustering. In nonfuzzy or hard clustering, data are divided into crisp clusters, where each data point belongs to exactly one cluster. In fuzzy clustering, the data points can belong to more than one cluster, and associated with each of the instances are membership grades that indicate the degree to which they belong to the different clusters.

Hard clustering techniques may be grouped into two categories: hierarchical and nonhierarchical [43]. A hierarchical clustering procedure involves the construction of a hierarchy or tree-like structure, which is basically a nested sequence of

partitions, while nonhierarchical or partitional procedures end up with a particular number of clusters at a single step.

1) Basic Algorithms: Nonhierarchical Techniques: The main nonhierarchical clustering techniques are: 1) k-means clustering and 2) self-organizing maps (SOM).

a) K-means clustering: The k-means clustering technique [58] is given as an input the number of clusters k. The algorithm then picks k items, called seeds, from the training set in an arbitrary way. Then, in each iteration, each input item is assigned to the most similar seed, and the seed of each cluster is recalculated to be the centroid of all items assigned to that seed. This process is repeated until the seed coordinates stabilize. This algorithm aims at minimizing an objective function J typically a squared error function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} d_{ij} = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \tag{1}$$

where d_{ij} is the distance measure between a data point x_i and the cluster center c_j. J is an indicator of the distance of the n data points from their respective cluster centers and represents the compactness of the clusters created.

Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The k-means algorithm is popular as it is easy to understand and implement. Its main drawback is its complexity that depends linearly on the number of patterns involved and on the number of clusters selected. Another problem is that it is sensitive to the initial partition—the selection of the initial patterns, and may converge to a *local* minimum of the criterion function value if the initial partition is not properly chosen. A possible remedy is to run the algorithm with a number of different initial partitions. If they all lead to the same final partition, this implies that the global minimum of the square error has been achieved. However, this can be time consuming, and may not always work.

b) SOM: The SOM algorithm was proposed by Teuvo Kohonen in 1981 [50]. Apart from being used in a wide variety of fields, the SOM offers an interesting tool for exploratory data analysis, particularly for partitional clustering and visualization. It is capable of representing high-dimensional data in a low-dimensional space [often a two-dimensional (2-D) or one-dimensional (1-D) array] that preserves the structure of the original data.

A self-organizing network consists of a set of input nodes $V = \{v_1, v_2, \ldots, v_N\}$, a set of output nodes $C = \{c_1, c_2, \ldots, c_M\}$, a set of weight parameters $W = \{w_{11}, w_{12}, \ldots, w_{ij}, \ldots, w_{\mathrm{NM}}\}$ ($1 \le i \le N$, $1 \le j \le M$, $0 \le w_{ij} \le 1$), and a map topology that defines the distances between any given two output nodes. The output nodes in SOM are usually arranged in a 2-D array to form a "map." Each input node is fully connected to every output node via a variable connection. A weight parameter is associated with each of these connections, and the weights between the input nodes and output nodes are iteratively changed during the learning phase until a termination criterion is satisfied. For each input vector $\boldsymbol{v}$, there is one associated *winner node* on the output map. A winner node is an output node that has minimum distance to the input vector.

The SOM algorithm starts by initializing the topology and size of the output map (M), the connection weights to random values over the interval [0,1], the gain value η (learning rate) and the neighborhood size r, and normalizes both the input vectors and the connected weight vectors. After that, the algorithm calculates the Euclidean distance between the new input vector $\boldsymbol{v}$ and each node on the output map, and designates the node with the minimum distance as the winner node c

$$c = \min \sqrt{\sum_{k=1}^{N} (v_k - w_{kj})^2}, \qquad j = 1, 2, \ldots, M. \tag{2}$$

Once c is obtained, the algorithm updates the weights W, the learning rate η, and N_c, the neighborhood surrounding the winner node c, in such way that the vectors represented by output nodes are similar if they are located in a small neighborhood. For each node $j \in N_c$, the SOM algorithm performs the following operation:

$$w_j^{(\mathrm{new})} = w_j^{(\mathrm{old})} + \eta[v_i - w_j^{(\mathrm{old})}]. \tag{3}$$

After that, the neighborhood size and the learning rate are decreased. This process is repeated until it converges. The neighborhood set N_c is a set of nodes that surround the winner node c. These nodes in N_c are affected with weight modifications, apart from those changes made to the winner node, as defined in the algorithm. These weight changes are made to increase the matching between the nodes in N_c and the corresponding input vectors. As the update of weight parameters proceeds, the size of the neighborhood set is slowly decreased to a predefined limit, for instance, a single node. This process leads to one of the most important properties of SOM that similar input vectors are mapped to geometrically close winner nodes on the output map. This is called neighborhood preservations, which has turned out to be very useful for clustering similar data patterns.

It is not always straightforward to visually inspect the projected data on the 2-D output map in order to decide the number and size of natural clusters. Therefore, careful analysis or postprocessing of output maps is crucial to the partition of the original data set. Like the k -means algorithm, the SOM produces a suboptimal partition if the initial weights are not chosen properly. Moreover, its convergence is controlled by various parameters such as the learning rate, the size, and shape of the neighborhood in which learning takes place. Consequently, the algorithm may not be very *stable* in that a particular input pattern may produce different winner nodes on the output map at different iterations.

2) Basic Algorithms: Hierarchical Techniques: The main problem of nonhierarchical approaches is that when working with high-dimensional problems, in general, there will not be enough items to populate the vector space, which will imply that most dimensions will be unreliable for similarity computations. In order to solve this problem, hierarchical clustering techniques were developed. There are two types of hierarchical clustering: agglomerative and divisive. Both share a common characteristic: they create a hierarchy of clusters. While the

agglomerative approach creates a bottom-up hierarchy, the divisive approach creates a top-down one. Generally speaking, divisive algorithms are computationally less efficient.

A typical hierarchical agglomerative clustering algorithm is outlined below.

Step 1) Place each pattern in a separate cluster.
Step 2) Compute the proximity matrix of all the interpattern distances for all distinct pairs of patterns.
Step 3) Find the most similar pair of clusters using the matrix. Merge these two clusters into one, decrement number of clusters by one, and update the proximity matrix to reflect this merge operation.
Step 4) If all patterns are in one cluster, stop. Otherwise, go to the above step 2).

The output of such an algorithm is a nested hierarchy of trees that can be cut at a desired dissimilarity level forming a partition. Hierarchical agglomerative clustering algorithms differ primarily in the way they measure the distance or similarity of two clusters where a cluster may consist of only a single object at a time. The most commonly used intercluster measures are

$$d_{AB} = \min_{\substack{i\in A\\ j\in B}}(d_{ij}) \tag{4}$$

$$d_{AB} = \max_{\substack{i\in A\\ j\in B}}(d_{ij}) \tag{5}$$

$$d_{AB} = \frac{1}{n_A n_B}\sum_{i\in A}\sum_{j\in B} d_{ij} \tag{6}$$

where d_{AB} is the dissimilarity between two clusters A and B, d_{ij} is the dissimilarity between two individual patterns i and j, n_A and n_B are the number of individuals in clusters A and B, respectively. These three intercluster dissimilarity measures are the basis of the three of the most popular hierarchical clustering algorithms. The *single-linkage* algorithm uses (4), the *minimum* of the distances between all pairs of patterns drawn from the two clusters (one pattern from each cluster). The *complete-linkage* algorithm uses (5), the *maximum* of all pairwise distances between patterns in the two clusters. The *group-average* algorithm uses (6), the average of the distances between all pairs of individuals that are made up of one individual from each cluster.

A challenging issue with hierarchical clustering is how to decide the *optimal* partition from the hierarchy. One approach is to select a partition that best fits the data in some sense, and there are many methods that have been suggested in the literature [30]. It has also been found that the single-linkage algorithm tends to exhibit the so-called *chaining* effect: it has a tendency to cluster together at relatively low level objects linked by chains of intermediates. As such, the method is appropriate if one is looking for "optimally" connected clusters rather than for homogeneous spherical clusters. The complete-linkage algorithm, however, tends to produce clusters that tightly bound or compact, and has been found to produce more useful hierarchies in many applications than the single-link algorithm [43]. The group-average algorithm is also widely used. Detailed discussion and practical examples of how these algorithms work can be found in [43] and [89].

3) Basic Algorithms: Fuzzy Clustering: One of the most widely used fuzzy clustering algorithms is the fuzzy C-means (FCM) algorithm [10]. The FCM algorithm attempts to partition a finite collection of elements $X = \{x_1, \ldots, x_n\}$ into a collection of c fuzzy clusters with respect to some given criterion. Given a finite set of data, the algorithm returns a list of c cluster centers $C = \{c_1, \ldots, c_c\}$ and a partition matrix $U = u_{i,j} \in [0,1], i = 1, \ldots n, j = 1, \ldots, c$, where each element tells the degree to which element x_i belongs to cluster c_j. Like the k-means algorithm, the FCM aims to minimize an objective function. The standard function is

$$J = \sum_{j=1}^{c}\sum_{i=1}^{n}(u_{i,j})^m \left\|x_i^{(j)} - c_j\right\|^2 \tag{7}$$

which differs from the k-means objective function by the addition of the membership values u_{ij} and the fuzzifier m. The fuzzifier m determines the level of cluster fuzziness. A large m results in smaller memberships u_{ij} and hence, fuzzier clusters. In the limit $m = 1$, the memberships u_{ij} converge to 0 or 1, which implies a crisp partitioning. In the absence of experimentation or domain knowledge, m is commonly set to 2. The basic FCM algorithm, given n data points $(x_1, \ldots, x_n)$ to be clustered, a number of c clusters with $c_1, \ldots, c_c)$ the center of the clusters, and m the level of cluster fuzziness with

$$m \in \mathbb{R} > 1 \tag{8}$$

first initializes the membership matrix U to random values, verifying that

$$u_{ij} \in [0,1], \sum_{j=1}^{c} u_{ij} = 1. \tag{9}$$

After the initialization, the algorithm obtains the center of the clusters $c_j, j = 1, \ldots, c$

$$c_j = \frac{\sum_{i=1}^{n}(u_{ij})^m x_i}{\sum_{i=1}^{n}(u_{ij})^m}. \tag{10}$$

and obtains the distance between all points $i = 1, \ldots, n$ and all cluster centers $j = 1, \ldots, c$

$$d_{ij} = \left\|x_i^{(j)} - c_j\right\|. \tag{11}$$

Updating matrix U according to the new distances

$$d_{ij} = 0 \Rightarrow u_{ij} = 1$$

$$u_{ij} = \left[\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{ik}}\right)^{\frac{2}{m-1}}\right]^{-1}. \tag{12}$$

This process is repeated until the set of cluster centers is stabilized. There are other algorithms, which are optimizations of the original FCM, like the fuzzy c-medoid algorithm (FCMdd) or the fuzzy c-trimered medoids algorithm (FCTMdd) [51].

4) Applications for UM: For UM, there are two kinds of interesting clusters to be discovered: usage clusters and page clusters. Clustering of users tends to establish groups of users exhibiting similar browsing patterns, which are usually called stereotypes. Such knowledge is especially useful for inferring user demographics in order to perform market segmentation in

TABLE I
EXAMPLES OF SOME CLUSTERING-BASED USER MODELS

	Application	Input Data	Results
Mobasher et al, 2000 [61]	Capture of web-users interests using k-means Clustering. Design of a cluster-based recommendation system	User logs from the Univ. of Minnesota Comp. Science web server collected during a month.	Example of the implementation of the recommendation system in a commercial site.
Paliouras et al. 1999 [66]	News- filtering system based on communities of users. Clustering allows to recommend interesting news to a user.	When registering a user specifies his/her interests.	Established machine learning techniques are very useful for the acquisition of communities of users
Doux et al., 1997 [27]	K-means clustering algorithm for user profiling in order to derive prototypical behavior from each user.	Data collected from a dedicated set of experiments where users are asked about their preferences.	The techniques proposed handle qualitative data for clustering users efficiently.
Fu et al., 1999 [34]	Grouping of users with a common behavior in a web server taking into account access patterns .	Data collected from UMR web server log (www.umr.edu) containing 2.5 million records.	The clusters obtained can be used for personalization purposes.
Hay et al., 2001 [39]	Clustering methods that capture the inherent sequentiality of web visits. A metric, Sequence Alignment Method, is introduced to be used instead of Euclidean distance for clustering purposes.	Log files of a Belgian telecom provider collected over a one-week period.	The results are as good as the ones obtained with Euclidean distance, while keeping the concept of order.
Mobasher et al., 2001 [62]	Clustering for collaborative filtering	12,000 sessions collected from the Association for Consumer Research web site.	With the proper data preprocessing the clustering approach outperforms more traditional approaches to this problem (like Nearest Neighbor)

e-commerce applications or provide personalized Web content to the users. However, clustering of pages will discover groups of pages having related content. This information is useful for Internet search engines and Web-assistance providers.

In the context of UM, clustering has a distinguishable characteristic: it is usually done with nonnumerical data. This implies that, usually, the clustering techniques applied are relational, where numerical values represent the degrees to which two objects of the data set are related. Clustering applied to UM has to use techniques that can handle relational data because the information used to create clusters (pages visited, characteristics of the user, etc.) cannot usually be represented by numerical vectors. In case they are represented using vectors, part of the semantic of the original data is lost. In these systems, the definition of distance is done using vectorial representations of user interactions with the AH system. Some examples of relational clustering applied to Web mining can be found in [47] and [61].

Table I summarizes some studies and applications of "hard" clustering for UM. Some examples of recommendation tasks implemented with clustering algorithms are presented in [27], [34], [61], and [62] and in [66] and [67], which uses SOM. Examples of classification tasks implemented using clustering are presented in [27] and [39]. Goren-Bar *et al*. [36] use SOM to classify documents based on a subjectively predefined set of clusters in a specific domain.

When using fuzzy clustering, a user can be at the same time in more than one cluster with different degrees of truth. This allows to better capture the inherent uncertainty that the problem of modeling user behavior has. Examples of applications that implement a recommendation task using FC include [52] and [65]. Examples of classification tasks are presented in [46] and [51]. Table II summarizes some studies and applications of FC for UM.

5) Limitations: The main problem that clustering techniques face is how to define the concept of distance that is to be used. In general, some knowledge of the problem is needed to define an optimum concept of distance. When applied to user modeling, this problem is even harder due to the nature of the data available: interactions, user preferences, pages visited, etc., which are not expressed in a numerical way. Different techniques to characterize Web-user behavior using numerical vectors have been proposed [46], [61], but in one way or the other, the representations lose part of the semantics that the original data had. More research is needed on how to express the data available in a numerical way and how to define a meaningful concept of distance for creating efficient user models using clustering techniques while at the same time keeping the semantics of the original data.

Nonhierarchical clustering techniques assume that the number of clusters k is known *a prioiri*. For UM, this is not usually the case. This implies that some heuristics need to be used to determine the number of clusters.

Also, user models developed so far using fuzzy clustering do not fully use the fuzzy nature of the technique in order to create more flexible and adaptive systems. More studies are needed to create meaningful ways of mixing the different personalizations associated with each one of the clusters in which a user can be included when using fuzzy clustering.

B. Association Rules for UM

Association rule discovery aims at discovering all frequent patterns among transactions. The problem was originally introduced by Agrawal in 1993 [1] and was based on detecting frequent items in a market basket. The *a priori* algorithm, including its variations and extensions, is widely accepted to solve this problem.

1) Basic Algorithms: Being $I = \{i_1, i_2, \ldots, i_m\}$ a set of items, S a set of transactions, where each transaction T is a set of items, and X a set of items in I, a transaction T is said to contain X, if

$$X \subseteq T. \tag{13}$$

TABLE II
EXAMPLES OF SOME FUZZY CLUSTERING-BASED USER MODELS

	Application	Input Data	Outcome
Lampinen and Koivisto , 2002 [52]	Obtain application profiles from network traffic data to manage network resources.	274000 samples of different applications from an edge router of a LAN network.	FCM produced better results than SOM. A method for the comparison of both solutions is also introduced.
Nasraoui et al. , 1999 [65]	A new algorithm (CARD) to mine user profiles from access logs is proposed.	12 day log data of the Dep. of Comp. Eng.. at Univ. of Missouri.	CARD is very effective for clustering many different profiles in user sessions.
Joshi et al. , 2000 [46]	Two algorithms to mine user profiles: FCMdd and FCTMdd.	CSEE logs of Univ. of Maryland	Both algorithms extract interesting user profiles. FCMdd is not able to handle noise as effectively as FCTMdd.
Krishnapuram et al. , 2001 [51]	Web access log analysis for user profiling using RFCMdd (Robust Fuzzy c-Medoids).	Five days of CSEE web server activity of Univ. of Maryland.	RFCMdd is very effective for clustering of relational data.

TABLE III
EXAMPLES OF SOME ASSOCIATION RULES-BASED USER MODELS

	Application	Input Data	Results
Chen et al., 2002 [20]	Prediction of user intention in machine-human interaction. The model is used to predict human actions for an Internet browser.	Interaction of 5 users with the system during a month.	The approach achieves 84% average accuracy in predicting user's intention.
Lin et al., 2001 [55]	Learning user preference models of multimedia Internet files. The model is used to predict when a user will need a multimedia file and to help in the search of the file needed.	200 records of user's emails.	The average correct acceptance and average correct refusal of each user preference modeling is higher that 80%.
Mobasher et al., 2001 [62]	Efficient personalization of web sites using association rules and non-labeled data contained in server's log file.	Access logs for the Web site Association for Consumer Research (ACR) Newsletter (www.acr-news.org).	Association Rules achieve better recommendation effectiveness that traditional collaborative filtering approaches.
Geyer-Schulz, 2002 [35]	Evaluation of the recommendations given in a educational Internet information broker using Association Rules and repeat-buying theory.	Data set obtained from the Virtual University Information system of Vienna University during 6 moths of usage.	Association Rules tend to work better and achieve higher prediction accuracy.
Nanopoulos et al, 2001 [64]	Association Rule discovery to construct a prediction system for effective prediction of web user request. Implementation of a prefetching system .	Clarknet trace log: http://ita.ee.lbl.gov/html/traces.html	The proposed algorithm WM_o outperforms the existing algorithms and is very efficient in reducing the number of candidates.

An association rule is defined as an implication of the form

$$X \Rightarrow Y \quad X \subset I \quad Y \subset I \quad X \cap Y = \emptyset \tag{14}$$

where X is usually called the antecedent and Y the consequent. The order of the items of both the antecedent and consequent is not relevant. Each association rule has a confidence and a support associated. The support of a rule is defined as the fraction of strings in the set of sessions of S where the rule successfully applies

$$\theta(XY) = \frac{|S_i \in S / XY \in S_i|}{|S|}. \tag{15}$$

The confidence of a rule is defined as the fraction of times for which if the antecedent X is satisfied, the consequent Y is also true

$$\sigma(XY) = \frac{\theta(XY)}{\theta(X)}. \tag{16}$$

The values of support and confidence are used to filter the set of association rules obtained from S. Given a set of transactions S, the task of mining association rules can be formulated as finding all association rules with at least minimum support and minimum confidence, where the thresholds for support and confidence are user-specified values. The task of discovering association rules can be done in three steps: 1) find all sets of items that have transaction support above minimum support; 2) for each item obtained, find all nonempty subsets; and 3) for each such subset X of item I, if the confidence is bigger than the given threshold, produce the rule

$$X \Rightarrow (I - X). \tag{17}$$

2) Applications for UM: In the context of UM, association rules capture sets of actions that have a causal relation among them. A typical application of association rules for UM is capturing pages that are accessed together.

Typically, association rules are used to implement recommendation tasks. Some examples of recommendation tasks implemented using association rules are presented in [20], [35], [55], [62], and [64]. Table III summarizes these examples.

3) Limitations: Association rules have two important drawbacks when used for UM: they do not capture the sequentiality of both the antecedent and the consequent and no temporality information (for example, when X happens when is Y going to happen) is captured. Both temporality and sequentiality are very important characteristics for UM. Temporality allows us to predict not only what actions a user is going to take, but also when those actions are going to be taken. Sequentiality captures the order in which a user makes a set of actions which is relevant to the consequent that such antecedent will fire. Because temporality and sequentiality are two key elements that need to be captured to create efficient user models, more research is needed to capture these characteristics when creating association rules.

IV. SUPERVISED APPROACHES TO UM

This section gives a review of how supervised learning techniques (decision trees/classification rules, neural networks, k-NN, and SVMs) can be used to model user behavior.

A. Decision Trees/Classification Rules for UM

Decision tree learning [60], [92] is a method for approximating discrete-valued functions with disjunctive expressions.

Decision tree learning is generally best suited to problems where instances are represented by attribute-value pairs and the target function has discrete output values.

Classification rules are an alternative representation of the knowledge obtained from classification trees. They construct a profile of items belonging to a particular group according to their common attributes.

1) Basic Algorithms: The training process that creates a decision tree is called induction. A standard decision tree algorithm has two phases: 1) tree growing and 2) pruning. The growing phase can be done using two methods: 1) top-down induction and 2) incremental induction.

Top-down induction is an iterative process which involves splitting the data into progressively smaller subsets. Each iteration considers the data in only one node. The first iteration considers the root node that contains all the data. Subsequent iterations work on derivative nodes that will contain subsets of the data. The algorithm begins by analyzing the data to find the independent variable that, when used as a splitting rule will result in nodes that are most different from each other with respect to the dependent variable. The quality of a test is measured by the impurity/variance of example sets. The most common measure is the information gain. Typically, the set of possible tests is limited to splitting the examples according to the value of a certain attribute. Once a node is split, the same process is performed on the new nodes, each of which contains a subset of the data in the parent node. This process is repeated until only nodes where no splits should be made remain.

Incremental induction is a method for the task of concept learning. When a new training example is entered, it is classified by the decision tree. If it is incorrectly classified, then the tree is revised. Restructuring the tree can be done by storing all training examples or by maintaining statistics associated with nodes in the tree.

Tree-building algorithms usually have several stopping rules. These rules are usually based on several factors including maximum tree depth, minimum number of elements in a node considered for splitting, or the minimum number of elements that must be in a new node.

The second phase of the algorithm optimizes the resulting tree obtained in the first phase. Pruning is a technique used to make a tree more general. It removes splits and the subtrees created by them. There is a great variety of different decision tree algorithms in the literature. Some of the more common algorithms are: classification and regression trees (CART) [15], [29], chi-squared automatic interaction detection (CHAID) [48], C4.5 [72], J4.8 [93], C5.0 [93] (which implements boosting), and ID3 [71].

Rules are, at its simplest form, an equivalent form of expressing a classification tree. In order to obtain the set of rules of a decision tree, each path is traced from the root node to the leaf node, recording the test outcomes as antecedents and the leaf-node classification as the consequent. The process of converting a decision tree into decision rules can be done after or before pruning.

Converting a decision tree to rules before pruning has some advantages: 1) it allows distinguishing among the different contexts in which a decision node is used; 2) the pruning decision regarding an attribute test can be made differently for each path; and 3) it improves readability. Nevertheless, obtaining the rules before pruning can produce a high number of rules. This can be solved by applying some rule-reduction techniques. The basic techniques are: 1) eliminate redundant and unnecessary antecedents; 2) use OR to combine rules in order to reduce the total number of rules to the number of dependent variables; and 3) eliminate unnecessary rules. Rules can be also obtained after the tree has been pruned, producing in this case a smaller rule knowledge base.

Algorithms such as CART, C4.5, and C.5 include methods to generalize rules associated with a tree which removes redundancies.

2) Applications for UM: In the context of UM, decision trees can be used to classify users and/or documents in order to use this information for personalization purposes. Decision trees can also handle noisy data and/or data with missing parameters, which makes them very useful for creating user models due to the noisy and imprecise nature of the data available. Table IV summarizes some studies and applications of decision trees for UM.

Classification trees are typically used to implement classification tasks. In this case, the classification trees are used to construct user models to personalize the user experience (for example, regarding their level of expertise, cognitive style, etc.) [7], [87]. Due to its ability to group users with similar characteristics, classification tress can be also used to implement recommendation tasks [66], [91], [96].

Classification rules are widely used to model user behavior as they provide a straightforward framework to represent knowledge. The readability of the output knowledge is a great advantage of this approach. Table V summarizes some studies and applications of classification rules for UM.

Examples of classification tasks implemented using classification rules can be found in [45], [82], [83], and [95]. In these cases, the rules are used to characterize different user stereotypes that are used to implement some personalized feature. Cercone *et al.* [18], [19] give examples of application of classification rules for recommendation.

Classification rules can not only be obtained from a learning algorithm but can also be directly expressed by the designer of the system. Some examples of this approach are [63] which constructs a model capturing the behavior of a user that is controlling a pole, [25] and [76], which use rules as part of the architecture of a general-purpose tool for adaptive Websites, and [9], which uses rules to characterize user behavior in the context of information retrieval.

3) Limitations: Decision trees/classification rules produce results that are highly dependent on the quality of the data available. This is because the subtrees are created using the maximum information gain possible. In some cases, if the information available is not appropriate, which typically happens when the information used to create user models has been obtained using user feedback or in a noisy environment, the models created will not correctly capture user behavior. Also, for high-dimensional problems, the response time of decision trees can be very high.

TABLE IV
EXAMPLES OF SOME DECISION TREE-BASED USER MODELS

	Application	Input Data	Results
Paliouras et al., 1999 [66]	Construction of user stereotypes using C4.5. Stereotypes are used to anticipate interests of a user within the context of a news retrieval system.	Questions answered by 31 users regarding the ECRAN information system.	It is very important to have good data in order to obtain good models.
Tsukada et al., 2001 [87]	Automatic classification of web pages in a pre-specified set of categories using C4.5 and association rules.	14 top categories of Yahoo! JAPAN. From each category 200 pages were randomly download.	The experimental evaluation demonstrates that this method provides acceptable accuracy with the classification of web-page into top categories of Yahoo! JAPAN.
Webb et al., 1997 [90]	Use of C4.5 to build the Feature Based Modeling instruction module. The results are applied to the Subtraction Modeler.	Test administered to 73 nine to ten year old primary school students. Each test contained 40 three column subtraction problem randomly generated.	C4.5 increases the number of predictions made without significantly altering the accuracy of those predictions.
Zhu et al., 2003 [196]	Construction of a recommender system to help users find relevant information on the web using C4.5 and Naïve Bayesian Classifier.	Collected data from 129 participants, asking each participant to perform two search tasks.	C4.5 outperforms Naïve Bayesian Classifier.
Beck et al., 2003 [7]	Construction of a User Model for an adaptive tutor with J4.8 and Naïve Bayesian classifier.	Data collected from the interaction of 88 students with the Reading Tutor.	Naïve Bayesian Classifier outperforms J4.8 for individual modeling and J4.8 outperforms Naïve Bayesian Classifier for Group modeling.

TABLE V
EXAMPLES OF SOME CLASSIFICATION RULES-BASED USER MODELS

	Application	Input Data	Results
Shah et al., 2002 [83]	Classification rules model bid strategies in an on-line auction website	Data collected from 2 auctions over different period of time, making a total of 11,537 bids collected.	The bidder model obtained can be used to simulate online markets and/or to detect fraud in on-line auction websites.
Zhang, 2003 [95]	Classification of the users of an information retrieval (IR) system according to different user stereotypes.	Ratings given by 64 people with four different backgrounds (stereotypes) regarding their IR knowledge.	The classification can be effectively used to personalize the information retrieval environment.
Joerding, 1999 [45]	Dynamic user modeling of the interests of a web shopper. The user model represents the present interest of that particular user.	Set of visits of a user to an adaptive shopping site.	Classification Rules efficiently model dynamic human behavior.
Cercone, 2002 [19]	ELEM2, a novel rule induction process.	Not Described	Applications to extract user profiles that are used to construct recommender systems.
Semeraro et al., 2001 [82]	Use of classification rules to model the user interaction in a digital library service architecture. Comparison between the results given by C4.5 and J4.8.	Log file containing the interactions of each user with the digital library system. The data is the preprocessed and labeled.	Accurate interaction models can be inferred automatically by using J4.8. J4.8 slightly outperforms C4.5.

This is inconvenient when working with adaptive systems, because real-time response is needed. This problem can be solved in some cases using classification rules.

The special interest in UM is the combination of classification rules with soft computing techniques (fuzzy logic and neural networks especially) in order to create more flexible user models. Fuzzy classification rules are able to overlap user models and to improve the interpretability of the results. So far, UM with fuzzy classification rules has not been used at its full capacity.

B. *k-NN for UM*

k-NN is a predictive technique suitable for classification [33]. Unlike other learning techniques in which the training data are processed to create the model, in k-NN, the training data represents the model.

1) Basic Algorithms: The simplest version is the nearest neighbor algorithm. In this case, the training portion of the algorithm simply stores the data points of the training data $S, S = \{x_1, \ldots, x_n\}$. To classify a new item i, the nearest-neighbor classifier finds the closest (according to some distance metric) training point x_j to i and assigns it to the same class of x_j. The nearest-neighbor algorithm is traditionally implemented by computing the distance from i to every x_j of S. It can also be implemented more efficiently using Voronoi diagrams.

The k-NN algorithm is a variation of nearest neighbor. In this case, instead of looking at only the point x_i that is closest to i, the algorithm looks at the k points in S that are closest to i. Since the class of each of the k nearest points is known, the majority category in this subset is the result of the classification. Nearest neighbor is a special case of k -nearest-neighbor, where $k = 1$.

k-NN has two main parameters: 1) the parameter k (number of nearest cases to be used) and 2) the distance metric d.

The selection of the metric is very important, because different metrics, used on the same training data, can result in completely different predictions. In general, the selection of the metric is application dependent. A typical metric d is the Euclidean distance, where

$$d(\mathbf{x}, \mathbf{i}) = \text{sqrt}((\mathbf{x} - \mathbf{i})^{\mathrm{T}}(\mathbf{x} - \mathbf{i})). \tag{18}$$

The value of k should be determined by experimentation. In general, higher values of k produce noiseproof systems, but also systems with higher computational costs. Typically, k is in the range of five to 20 rather than in the order of hundreds.

Because the model created by k-NN is basically the set of training points, this means, especially for high-dimensional problems and/or for problems with a lot of training data, that the amount of memory needed to store the model is very high, and that it increases when more data are added to the system.

TABLE VI
EXAMPLES OF SOME k-NN BASED USER MODELS

	Application	Input Data	Results
Billsus et al, 1999 [12]	Construction of an intelligent agent designed to compile a daily news program for individual users using a user model that captures user interests using a multi-strategy approach.	Training data of the agent by 10 users during 4-8 days. K-NN is used to model Short Term interest of a user.	A multi-strategy learning approach (a short-term model and a long-term model) captures changes in user interest very efficiently.
Schwab et al., 1999 [81]	ELFI (Electronic Funding Information) a WWW system that provides personalized information about research funding.	ELFI log file of several months of system usage. The selection of a program is regarded as interesting for that user.	The optimum solution is obtained with a combination of K-NN and Naïve Bayesian Classifiers.
Tsiriga et al., 2002 [86]	Initialization of the Student Model in a Algebra Tutor using k-NN and stereotypes based on recognized similarities with other students that have already interacted with the system.	Information is captured by the Algebra Tutor using a set of questions to the new user of the system. This information is used to obtain the stereotype.	The combination of K-NN and stereotypes produces very good results for Student Modeling.
Shih et al., 2001 [85]	Development of an on-line learning system using k-NN for providing adaptive learning materials on a course of introduction to computer networks.	Not Described	Application of k-NN compared with previous researches can retrieve more adequate materials based on previous similar cases.

Agglomerative nearest neighbor (A-NN) algorithms have been proposed to solve this problem. A-NN is designed to capture the same information as k-NN but without the necessity of storing the complete set of training data. The key idea behind A-NN is that it clusters training points that have the same class. Each class is then represented by a class representative, against which the concept of distance is measured when classifying a new instance. A-NN has been traditionally used for collaborative clustering.

2) Applications for UM: k-NN algorithms can be used very effectively for recommendation. In this case, the algorithm is used to recommend new elements based on the user's past behavior. A typical application is the recommendation of interesting documents. As can be seen, the concept of distance (in this case, the distance between two texts) is the key element to an efficient recommendation system. Examples of k-NN used to implement recommendation tasks are presented in [12], [81], and [85].

Nevertheless, like any classification algorithm, it can also be used to implement classification tasks, for example to initialize properly a new user when first visiting a system given a set of existing user models. Tsiriga and Virvou [86] give an example of k-NN used to implement a classification task. Table VI summarizes some studies and applications of k-NN for UM.

3) Limitations: k-NN is a very simple technique that produces good and straightforward results. Nevertheless, the main drawbacks of this approach are the high dependence of the results on the value of k and the metric d chosen. In general, to choose both parameters, some experimentation with the system and some knowledge on how the system works will be needed. Also, the value k should be big enough to produce a classification system noiseproof; otherwise a k-NN with a small k value will be extremely susceptible to noise. In the context of UM, this is very important because usually the data available is noisy and imprecise due to the nature of the problem. Nevertheless, an increase in the value of k implies an increase in the computational time needed. The definition of the distance d, when used for UM, faces the same problems as clustering techniques: 1) the necessity of transforming user-related information into a vectorial representation without losing semantics and 2) the definition of a concept of distance among those vectors that capture the characteristics of the problem (user interest, user behavior, etc.). Another limitation of k-NN for UM is the response time, which is directly affected by d and by the dimensionality and number of examples of the training data. AH systems need real-time response, and k-NN may not be suitable in some cases.

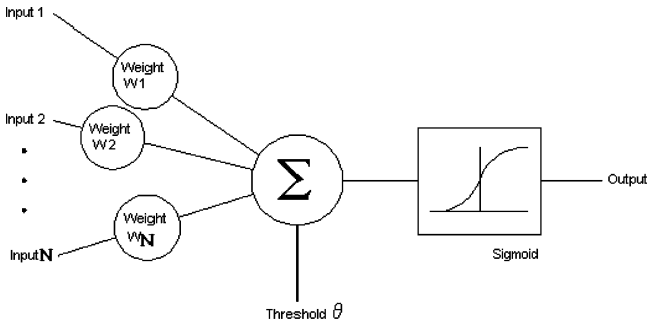

Fig. 3. Architecture of an artificial neuron.

C. Neural Networks for UM

An artificial neural network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems process information [31], [40]. Although typical ANNs are designed to solve supervised problems, there are also architectures to solve unsupervised problems.

1) Basic Concepts: The key element of this paradigm is the structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in parallel. They consist of the following elements: 1) neurones; 2) weighted interconnections; 3) an activation rule, to propagate signals through the network; and 4) learning algorithm, specifying how weights are adjusted.

The basic element of any ANN is an artificial neuron (Fig. 3). A neuron has N weighted input lines and a single output. The neuron will combine these weighted inputs by forming their sum and, with reference to an activation function and a threshold value; will determine its output.

Here $x_1, x_2, \ldots, x_N$ are the input signals, $w_1, \ldots, w_N$ the synaptic weights, u the activation potential, θ the threshold, y the output signal, and f the activation function

$$u = \Sigma_{i=1}^{N} w_i x_i \tag{19}$$

$$y = f(u - \theta). \tag{20}$$

TABLE VII
EXAMPLES OF SOME NN-BASED USER MODELS

	Application	Input Data	Results
Bidel et al., 2003 [11]	Classification and tracking of user navigation.	Data generated from an on-line encyclopedia.	A labeled approach to the problem produces better accuracy.
Sas et al., 2003 [79]	Prediction of user's next step in a virtual environment	30 users performed exploration and searching within the environment.	Very accurate predictions of the next step
Shepperd, 2002 [84]	Adaptive filtering system for electronic news using stereotypes.	The Halifax Herald Ltd.	Very useful for readers with specific information needs.
Beck and Woolf., 1998 [8]	Construction of a student model for an intelligent tutoring system.	Data collected by the tutoring system	NN-based recommendation to each individual.

TABLE VIII
EXAMPLES OF SOME SVM-BASED USER MODELS

	Application	Input Data	Results
Liao, 2002 [54]	Profiling user behavior in Windows NT using CPU time to detect a misuse or an attack on the system.	Goldring's Windows NT Profiling dataset. A user login is then transformed into a vector containing CPU time of each process.	The system identifies correctly each user if the set of users is small (5 users). For bigger sets of users (10 users) the classification rate is not as satisfactory.
Ruvini, 2003 [77]	A front-end to Google search engine that uses SVM to infer the user's goal and filter the set of links in order to avoid the problem of screen size in mobile devices.	Recorded users interactions with Google in a predetermined set of searching tasks.	The problem of small screen sizes is satisfactorily solved.
Aihara et al., 2001 [2]	Interactive document retrieval system supported by a personalization of a basic classification of categories that is adapted to each user using SVM.	Not Described	SVM effectively adapt the boundaries of each category according to each user interests.

Defining $w_0 = \theta$ and $x_0 = -1$, the output of the system can be reformulated as

$$y = f\left(\Sigma_{i=0}^{N} w_i x_i\right). \tag{21}$$

The activation function f defines the output of the neuron in terms of the activity level at its input. The most common form of activation function used is the sigmoid function.

There are very different ways in which a set of neurons can be connected among themselves. The traditional cluster of artificial neurons is called a neural network. Neural networks are basically divided into three layers: the input layer, the hidden layer, which may contain one ore more layers, and the output layer.

The layer of input neurons receives the data either from input files or directly from electronic sensors in real-time applications. The output layer sends information directly to the outside world, to a secondary computer process, or to other devices such as a mechanical control system. Between these two layers can be many hidden layers. These internal layers contain many of the neurons in various interconnected structures. The inputs and outputs of each of these hidden neurons simply go to the other neurons. In most networks, each neuron in a hidden layer receives the signals from all of the neurons in a layer above it, typically an input layer. After a neuron performs its function, it passes its output to all of the neurons in the layer below it, providing a feed-forward path to the output. Another type of connection is feedback. This is where the output of one layer routes back to a previous layer.

Multilayer perceptrons (MLPs) are the typical architecture of NNs. MLPs are full-connected feedforward nets with one or more layers of nodes between the input and the output nodes.

Classification and recognition capabilities of NNs stem from the nonlinearities used within the nodes. A single-layered perceptron implements a single hyperplane. A two-layer perceptron implements arbitrary convex regions consisting of intersection of hyperplanes. A three-layer NN implements decision surfaces of arbitrary complexity [56], [57]. That is the reason why a three-layer NN is the most typical architecture.

NNs learn through an iterative process of adjustments. There are two training approaches: supervised and unsupervised.

In supervised training, both the inputs and the outputs are provided. The net is trained by initially selecting small random weights and internal thresholds, and presenting all training data repeatedly. Weights are adjusted after every trial using information specifying the correct class until weights converge and the cost function is reduced to an acceptable value. The vast bulk of networks utilize supervised training. The most common supervised technique is the backpropagation learning algorithm. It uses a gradient search technique to minimize a cost function defined by the mean square error (MSE) between the desired and the actual net outputs, with l the number of training points

$$\text{MSE} = \frac{1}{l}\Sigma_{i=1}^{l}(y_i - y_i^{\wedge})^2. \tag{22}$$

The generally good performance found for the backpropagation algorithm is somewhat surprising considering that it is a gradient descent technique that may find a local minimum in the cost function instead of the desired global minimum.

2) Applications for UM: NNs are able to derive meaning from complicated and/or imprecise data. Also, NNs do not require the definition of any metric (unlike k-NN or clustering) which make them completely application independent. No initial knowledge about the problem that is going to be solved is needed. These characteristics make NNs a powerful method to model human behavior and an ideal technique to create user models for hypermedia applications.

NNs have been used for classification and recommendation in order to group together users with the same characteristics and create profiles and stereotypes. Bidel *et al.* [11] give an example of NNs used for classification, [7], [8], [79], and [84] use NNs for recommendation tasks. Table VII presents more details of these applications.

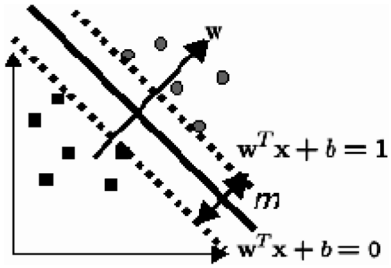

Fig. 4. SVM maximization of the distance between classes.

3) Limitations: NNs have been successfully used for UM mainly because they do not need any heuristic to produce a model. Nevertheless, they still face important limitations. The main ones are the training time needed to produce a model (which in cases can be measured in the order of many hours and even days) and the amount of information needed. The training time is an inconvenience for creating dynamic models. Although there are techniques that are able to retrain NNs dynamically, the techniques used so far for UM retrain the system from scratch in case more information, e.g., a new user or a new document, is added. More research in the field of incremental learning is needed. Another important limitation of NNs is their black box behavior. While the previous techniques, to a different extent, can be interpreted and manually changed, NNs cannot be interpreted which limits their applications in case a human understandable user model is needed.

D. SVMs for UM

SVM is a classifier derived from statistical learning theory [14], [23], [81]. The main advantages of SVM when used for classification problems are: 1) the ability to work with high-dimensional data and 2) high generalization performance without the need to add *a priori* knowledge, even when the dimension of the input space is very high.

The problem that SVM try to solve is to find an optimal hyperplane that correctly classifies data points and separates the points of two classes as much as possible. Fig. 4 isan example of the previous situation.

1) Basic Algorithms: Given two classes, the objective is to identify the hyperplane that maximizes m

$$m = \frac{2}{\|\mathbf{w}\|} \tag{23}$$

while at the same time classifying correctly all the examples given. Being $\mathbf{w}^{\mathrm{T}}$ the hyperplane that verifies the previous condition, all points that were part of that class will verify $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b > 0$, where $\mathbf{x}$ is the point that is being validated. If a point is not part of that class, then $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b < 0$.

Formally, the problem can be presented as follows. Given d the dimension of the problem and N the number of training points, let

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \Re^d \times Y, y_i \in Y, Y = \{-1, 1\} \tag{24}$$

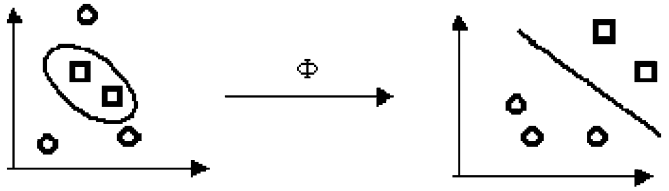

Fig. 5. Transformation of a nonlinearly separable problem into a linearly separable problem.

be the set of labeled inputs, where -1 indicates that the input is not of that class and 1 indicates that the input is of that class. The decision boundary should verify

$$\begin{aligned} \mathbf{x}_i^{\mathrm{T}}\mathbf{w}^{\mathrm{T}} + b \geq 1, \quad \forall y_i = 1 \\ \mathbf{x}_i^{\mathrm{T}}\mathbf{w}^{\mathrm{T}} + b \geq -1, \quad \forall y_i = -1. \end{aligned} \tag{25}$$

The problem can be presented as

$$\text{Maximize } m = \frac{2}{\|\mathbf{w}\|}$$

$$\text{Subject to } y_i(\mathbf{x}_i\mathbf{w}^T + b) \geq 1, \quad \forall i = 1, \ldots, n. \tag{26}$$

This formulation can be expressed as a standard quadratic problem (QP) in the form of

$$\text{maximize} \sum_{\mathrm{i}}^{\mathrm{N}} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j$$

$$\text{Subject to } \alpha_{\mathrm{i}} \geq 0, \sum_{i=1}^{N} \alpha_i y_i = 0. \tag{27}$$

In this context, a global maximum α_i can always be found and $\mathbf{w}$ can be recovered as

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i. \tag{28}$$

Many of the α_i are zero, which implies that $\mathbf{w}$ is a linear combination of a small number of data. The set of elements $\mathbf{x}_i$ with nonzero α_i are called support vectors. Graphically, these are the set of points that mark the border of the class.

This approach is valid whenever the set of points of the two classes are linearly separable. Nevertheless, in real data, this is usually not the case. In order to work with nonlinear decision boundaries the key idea is to transform $\mathbf{x}_i$ to a higher dimension space, so that in this new space the samples can be linearly divided (Fig. 5). The problem of this solution can be the high computational requirements. SVM solve these problems using kernels. The relationship between the kernel function K and Φ is

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j). \tag{29}$$

Intuitively, $K(\mathbf{x}, \mathbf{y})$ represents our desired notion of similarity between data $\mathbf{x}$ and $\mathbf{y}$ based on our prior knowledge of the problem. $K(\mathbf{x}, \mathbf{y})$ needs to satisfy a technical condition (Mercer condition) in order for Φ to exist. In practice, K can be directly specified, thereby specifying Φ indirectly, instead of choosing Φ. An example of a kernel function is the Gaussian kernel, which

TABLE IX
SELECTION OF SUITABLE DATA MINING TECHNIQUE

	Labeled Data		Unlabelled Data	
Task	**Readability Needed**	**Readability Not Needed**	**Readability Needed**	**Readability Not Needed**
Recommendation	Decision Trees Classification Rules	Neural Networks k-NN	Association Rules SOM	K-means Clustering Fuzzy Clustering
Classification	Decision Trees Classification Rules	SVM Neural Networks k-NN	SOM	K-means Clustering Fuzzy Clustering

is defined as

$$K(\mathbf{x_i}, \mathbf{x_j}) = e^{\frac{-\|\mathbf{x_i}-\mathbf{x_j}\|^2}{\mathrm{std}^2}}. \tag{30}$$

When working with a Gaussian kernel, std represents the standard deviation, and ideally should represent the minimum distance between any two elements of two different classes. As can be seen, when constructing a SVM based on a Gaussian kernel, the only value that needs to be defined is std. There are other standard kernels like polynomial, linear, or sigmoidal, each one with its own parameters. These kernels are generic in the sense that they can, in some way or the other, be applied to all problems and will provide good solutions. For example, Gaussian kernels tend to work very well in image recognition problems. Nevertheless, results can always be improved by designing a problem-specific kernel, although this approach requires an *a -priori* knowledge about the problem.

When working with kernels, in general it would not be possible to obtain $\mathbf{w}$. Nevertheless, SVM can still be used. Being N_S the number of support vectors of the training set, the decision function can be expressed as

$$f(x) = \mathrm{Sign}\left(\sum_{i=1}^{N_S} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right). \tag{31}$$

Although the theoretical background given has introduced classification system for only two classes, SVM can be generalized to a set of C classes. In this case, each one of the classes will be trained against the rest $C-1$ classes, reducing the problem to a two-class classification problem.

2) Applications for UM: Due to its ability to work extremely well with high-dimensional data, SVM can be used to classify documents in order to personalize recommendations given by search engines. Aihara and Takasu [2], Liao [54], and Ruvini [77] give examples of classification tasks implemented using SVM. Table VIII summarizes some studies and applications of SVM for UM.

3) Limitations: In general, the main problem that SVM faces is the selection of the kernel K used to implement the classifier. Although there are a set of standard kernels (polynomial, Gaussian, etc.) that work well for many applications, for complex cases it is convenient to develop a specific kernel considering the characteristics of the problem. For example, there are kernels specifically designed for gene detection or for text classification.

In the context of UM, SVM has been used to personalize recommendation applications. So far the classification has been applied to the documents but not to the users directly. Due to the ability of SVM to optimize the separation between the classes, the application of SVM to create classes of users can improve the individual degree of satisfaction of each user. More research is needed in the applications of SVM for UM, ranging from general applications to the development of specific kernels.

V. CRITERIA FOR THE SELECTION OF THE TECHNIQUES

The preceding discussion has demonstrated that each technique captures different relationships among the data available and expresses it using different data structures. In this section we present guidelines to help decide which technique to use when developing an AH application.

We consider that, in the context of UM, there are three main criteria that determine which data mining/machine learning technique is suitable for a specific adaptive application: 1) the labeled/unlabelled nature of the data available; 2) the type of task that is going to be implemented (recommendation or classification); and 3) the "readability" needed for the results. "Readability" is defined as the ability of a technique to produce a human-readable output of the knowledge captured for a non-technical user. There are two possible values for readability: 1) needed; and 2) not needed. The first one expresses the necessity of having a human-readable output while the second one states that this factor is not relevant. Table IX presents a guideline of which data mining techniques are useful considering the criteria previously introduced. The techniques are classified according to the set of references used in this study. The set of techniques that can be applied when the systems needs readability can also be applied when this factor is not relevant.

In general, when selecting a data mining technique two of the more important factors are: 1) the ability to handle high-dimensional data and 2) scalability. Although the ability to handle high-dimensional data for a generic problem is a very important characteristic, within the context of UM we consider that it is not a very relevant factor because of the dimensionality of the data used. Nevertheless, in the context of UM, the scalability of the techniques is a very important factor due to the high number of users that, in general, will interact with an AH system. The scalability of each technique regarding the number of users will depend on how the information of each user is presented. An indication of the scalability of each technique is presented in the first column of Table X.

Table X summarizes the characteristics of the techniques presented along four dimensions. The first three dimensions capture some of the main problems that machine learning for UM faces [91]: Computational complexity for off-line processing;

TABLE X
GENERAL CHARACTERISTICS OF THE REVISED TECHNIQUES

	Off-Line Complexity (Indication of Scalability)	Dynamic Modeling	Labeled / Unlabeled	Readability
K-means Clustering	$O(k \cdot m \cdot n \cdot i)$ [38] n number of instances to cluster m number of attributes k number of clusters i number of iterations, with $i=O(n)$ [24]	No	Unlabeled	No
SOM	$O(n)$, with n the number of feature vectors [74] $O(M^2)$ where M is the number of map units: each learning step requires $O(M)$ and to achieve a sufficient statistical accuracy the number of iterations should be at least some multiple of M. [47]	No	Unlabeled	Yes
Fuzzy Clustering	$O(n^2)$ with n the number of objects For some optimized algorithms $O(nlogn)$ [51]	No	Unlabeled	No
Association Rules	NP-Complete Exponential with the number of items [6]	No	N/A	Yes
Decision Trees	For single attribute, multi-way splits on A discrete variables and data size of N: $O(A^2N)$ For continuous attributes: $O(A^2N^3)$ [59]	Yes	Labeled	Yes
Classification Rules	Same as Decision Trees + Rule generation	Yes	Labeled	Yes
k-NN	$O(t\ c)$, with c the number of clusters and t the number of training samples. Empirical sample complexity is exponential in the number of irrelevant features [53]	Yes	Labeled	No
Neural Networks	NP-Complete for a generic 3 layer NN [13] Polynomial for some simple two layer networks [13]	Yes	Both	No
SVM	Complexity of Solving a Quadratic Optimization problem (QP) at each iteration: $O(N^3)$ with N total number of training points. [21] In general it is highly dependent of the SVM implementation used. [21]	No	Labeled	No

dynamic modeling, which indicates the suitability of the technique to change a user model on-the-fly; and labeled/unlabeled. The "readability" dimension has also been added to the table.

The combination of Table IX and X can be used to guide a choice of which technique to use when modeling user behavior for adaptive systems. First, Table IX identifies the set of techniques suitable for the adaptive application, and, after that, Table X can be used to refine the choice considering the scalability and dynamic modeling capabilities of each technique.

Finally, there are other techniques not reviewed in this paper, mainly predictive statistical techniques [97], that can also be used to create user models. For example, recommendation tasks have also been implemented with Markov models [3], [4], [5], [26], [28], [78] or with Bayesian networks [22], [37], [41], [44], [80], [94]. Classification tasks have also been implemented with Bayesian classifiers [20], [81].

VI. CONCLUSION AND DISCUSSION

Hypermedia systems are becoming more important in our everyday activities and their contents and services are evolving. Due to this important role, users welcome any improvement on the hypermedia services they receive. One trend used to improve digital services is through personalization which is based on the concept of UM. In this context, data mining and machine learning techniques can be used to automatically identify user patterns and interests and use the knowledge obtained to produce user models.

This paper has presented a review of the state of the art of data mining techniques within the area of UM for AH systems. The review demonstrates that one of the main problems that the development of user model faces is the lack of any kind of standardization for the design of user models. In order to improve this situation, this paper has tried to give a set of guidelines that formalize the design of user models using a data mining approach. In the same line of standardization, another interesting research area would be the design of a system to evaluate the performance of UM in order to make it feasible to compare different user models.

It is our opinion that the future of UM is in hybrid systems. As has been shown, each technique captures different elements of user behavior. The combination of these techniques among themselves and with other machine learning techniques, especially with soft computing techniques, will provide a useful framework to efficiently capture the natural complexity of human behavior.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," presented at the 1993 ACM SIGMOD Conf., Washington, DC, 1993.

[2] K. Aihara and A. Takasu, "Category based customization approach for information retrieval," in *8th Int. Conf. User Modeling, Lecture Notes in Artificial Intelligence*. vol. 2109, Berlin, Germany: Springer-Verlag, 2001.

[3] D. Albrecht, I. Zukerman, and A. Nicholson, "Pre-sending documents on the WWW: A comparative study," in *Proc. 16th Int. Joint Conf. Artif. Intell*., 1999, pp. 1274–1279.

[4] C. R. Anderson, P. Domingos, and D. S. Weld, "Adaptive Web navigation for wireless devices," presented at the 17th Int. Joint Conf. Artifical Intelligence, Seattle, WA, 2001.

[5] C. Anderson, P. Domingos, and D. Weld, "Relational Markov models and their application to adaptive Web navigation," in *Proc. 8th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2002, pp. 143–152.

[6] F. Angiulli, G. Ianni, and L. Palopoli, "On the complexity of mining association rules," *Data Mining Knowl. Discovery*, vol. 2, no. 3, pp. 263–281, 1998.

[7] J. Beck, P. Jia, J. Sison, and J. Mostow, "Predicting student help-request behavior in an intelligent tutor for reading," in *Proc. 9th Int. Conf. User Model.,* Lecture Notes in Artificial Intelligence, vol. 2702, Berlin, Germany: Springer-Verlag, 2003, pp. 303–312.

[8] J. E. Beck and B. P. Woolf, "Using a learning agent with a student model," in *Lecture Notes in Computer Science*. vol. 1452, Berling, Germany: Springer-Verlag, 1998, pp. 6–15.

[9] E. Benaki, V. A. Karkaletsis, and C. D. Spyropoulos, "User modeling in WWW: The UMIE prototype, adaptive systems and user modeling on the World Wide Web," presented at the 6th Int. Conf. User Modeling, Sardinia, Italy, Jun. 1997.

[10] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[11] S. Bidel, L. Lemoine, F. Piat, T. Artieres, and P. Gallinari, "Statistical machine learning for tracking hypermedia user behavior," presented at the 2nd Workshop Machine Learning, Information Retrieval, and User Modeling, 9th Int. Conf. User Modeling, Pittsburgh, PA, 2003.

[12] D. Billsus and N. Pazzani, "A hybrid user model for news story classification," in *Proc. 7th Int. Conf. User Modeling*, 1999, pp. 99–108.

[13] A. L. Blum and R. L. Rivest, "Training a 3-node neural network is NP-complete," *Neural Netw.*, vol. 5, no. 1, pp. 117–127, 1992.

[14] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Computational Learning Theory*, 1992, pp. 144–152.

[15] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees* (Wadsworth International Group/Probability Series). Belmont, CA: Wadsworth, 1984.

[16] P. Brusilovsky and E. Schwarz, "User as student: towards an adaptive interface for advanced Web-based applications," in *User Modeling: Proc. 6th Int. Conf.*, A. Jamesson, C. Paris, and C. Tasso, Eds. New York: Springer Wien, 1997.

[17] J. Callan, A. Smeaton, M. Beaulieu, P. Borlund, P. Brusilovsky, M. Chalmers, C. Lynch, J. Riedl, B. Smyth, and U. Straccia, "Personalization and recommender systems in digital libraries," presented at the 2nd DELOS Workshop Personalization and Recommender Systems in Digital Libraries, Dublin, Ireland, 2001.

[18] N. Cercone, L. Hou, V. Keselj, A. An, K. Naruedomkul, and X. Hu, "From computational intelligence to Web intelligence," *Computer*, vol. 35, no. 11, pp. 72–76, Nov. 2002.

[19] N. Cercone, "From computational intelligence to Web intelligence: An ensemble from potpourri," in *Proc. Web Intelligence Research and Development,* Lecture Notes in Artificial Intelligence, vol. 2198, Berlin, Germany: Springer-Verlag, 2002, pp. 31–42.

[20] A. Chen, F. Lin, W. Ma, and L. Wenyin, "User intention modeling in Web applications using data mining," *World Wide Web: Internet Web Inf. Syst.*, vol. 5, pp. 181–191, 2002.

[21] K. K. Chin, "Support vector machines applied to speech pattern classification" M. Phil. thesis, Dept. Eng., Cambridge Univ., 1998.

[22] C. Conati, A. Gertner, K. Vanlehn, and M. Druzdzel, "On-line student modeling for coached problem solving using Bayesian networks," presented at the 1997 User Modeling Conf., Sardinia, Italy, 1997.

[23] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[24] I. Davidson and A. Satyanarayana, "Speeding up k-means clustering by Bootstrap averaging," presented at the IEEE Data Mining Workshop on Clustering Large Data Sets, 3rd IEEE Int. Conf. Data Mining, Melbourne, FL, 2003.

[25] P. DeBra and N. Stash, "AHA! A general-purpose tool for adaptive Websites," presented at the World Wide Web Conf., Honolulu, HI, May 2002.

[26] M. Deshpande and G. Karypis, "Selective Markov models for predicting Web-page accesses," presented at the SIAM Int. Conf. Data Mining (SDM2001), San Francisco, CA.

[27] A. Doux, J. Laurent, and J. Nadal, "Symbolic data analysis with the k-means algorithm for user profiling, user modeling," presented at the 6th Int. Conf., UM97, Sardinia, Italy, 1997.

[28] D. Duchamp, "Prefetching hyperlinks," in *Proc. 2nd USENIX Symp. Internet Technologies and Systems,* USENIX, Oct. 1999, pp. 127–138.

[29] B. Efron and R. Tibshirani, "Statistical data analysis in the computer age," *Science*, vol. 253, no. 5018, pp. 390–395, 1991.

[30] B. S. Everitt, *Cluster Analysis*, 3rd ed. London, U.K.: Arnold, 1993.

[31] L. Fausett, *Fundamentals of Neural Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[32] J. Fink, A. Kobsa, and A. Nill, "Adaptable and adaptive information access for all users, including the disabled and the elderly," in *User Modeling: Proc. 6th Int. Conf.*, UM97, A. Jamesson, C. Paris, and C. Tasso, Eds. New York: Springer, 1997, pp. 171–173.

[33] J. H. Friedman, F. Baskett, and L. J. Shustek, "An algorithm for finding nearest neighbors," *IEEE Trans. Comput.*, vol. 24, no. 10, pp. 1000–1006, Oct. 1975.

[34] Y. Fu, K. Sandhu, and M.-Y. Shih, "Clustering of Web users based on access patterns," presented at the 1999 KDD Workshop Web Mining, San Diego, CA, Jul. 2002.

[35] A. Geyer-Schulz and M. Hahsler, "Evaluation of recommender algorithms for an Internet information broker based on simple association rules and on the repeat-buying theory," in *Proc. WEBKDD 2002, 4th WEBKDD Web Mining Usage Patterns User Profiles*, 2002, pp. 100–114.

[36] D. Goren-Bar, T. Kuflik, D. Lev, and P. Shoval, "Automatic personal categorization using artificial neural networks," in *Proc. 8th Int. Conf. User Modeling,* Lecture Notes in Artificial Intelligence, vol. 2109, Berlin, Germany: Springer-Verlag, 2001, pp. 188–198.

[37] B. Grobmann-Hutter, A. Jameson, and F. Witting, "Learning Bayesian networks with hidden variables for user modeling," presented at the Workshop Machine Learning for User Modeling, 6th Int. Conf. User Modeling, Sardinia, Italy, 1997.

[38] J. Hartigan, *Clustering Algorithms*. New York: Wiley, 1975.

[39] B. Hay, G. Wets, and K. Vanhoof, "Clustering navigation patterns on a Website using a sequence alignment method," presented at the IJCAI 2001 Workshop Intelligent Techniques for Web Personalization, Seattle, WA.

[40] S. Haykin, *Neural Networks, 2nd ed.* Englewood Cliffs, NJ: Prentice-Hall, 1999.

[41] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, "The Lumière project: Bayesian user modeling for inferring the goals and needs of software users," in *Uncertainty in Artificial Intelligence: Proc. 14th Conf.*, G. Cooper and S. Moral, Eds., 1998, pp. 256–265.

[42] A. Jain and R. C. Dubes, *Algorithms for Clustering Data*.. Englewood Clifs, NJ: Prentice-Hall, 1988.

[43] A. Jain and R. C. Dubes, "Data clustering," *ACM Comput. Survey*, vol. 31, pp. 264–323, 1999.

[44] F. V. Jensen, *An Introduction to Bayesian Networks*. Berlin: Springer-Verlag, 1996.

[45] T. Joerding, "Temporary user modeling approach for adaptive shopping on the Web," presented at the 2nd Workshop Adaptive Systems and User Modeling on the WWW, Toronto, Canada, 1999.

[46] A. Joshi, K. Joshi, and R. Krishnapuram, "On mining Web access logs," in *Proc. ACM-SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery*, 2000, pp. 63–69.

[47] S. Kaski, "Data exploration using self organizing maps," *Acta Polytech. Scand., Math., Comput. Manag. Eng. Ser.*, vol. 82, 1997.

[48] G. V. Kass, "An exploratory technique for investigating large quantities of categorical data," in *Appl. Stat.*, vol. 29, 1980, pp. 119–127.

[49] A. Kobsa, "Generic user modeling systems," in *Proc. User Modeling and User-Adapted Interaction.*, vol. 11, 2001, pp. 49–63.

[50] T. Kohonen, *Self-Organizating Maps*. New York: Springer-Verlag, 1997.

[51] R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi, "Low-complexity fuzzy relational clustering algorithms for web mining," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 4, pp. 595–608, Aug. 2001.

[52] T. Lampinen and H. Koivisto, "Profiling network applications with fuzzy C-means clustering and self-organising map," presented at the Int. Conf. Fuzzy Systems and Knowledge Discovery, Singapore, Nov. 2002.

[53] P. Langley and W. Iba, "Average-case analysis of a nearest neighbor algorithm," in *Proc. 13th Int. Joint Conf. Artificial Intelligence*, 1993, pp. 889–894.

[54] Y. Liao, "Windows NT user profiling with support vector machines," presented at the 2002 UC Davis Student Workshop on Computing, Technical Report CSE-2002-28, Department of Computer Science, University of California, Davis, 2002.

[55] F. Lin, L. Wenyin, Z. Chen, H. Zhang, and T. Long, "User modeling for efficient use of multimedia files," in *Proc. Advances in Multimedia Information Processing PCM 2001: 2nd IEEE Pacific Rim Conf. Multimedia*, 2001, pp. 24–26.

[56] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, pp. 4–28, 1987.

[57] C. G. Looney, *Pattern Recognition Using Neural Networks: Theory and Algorithms for Engineers and Scientists*. London, U.K.: Oxford Univ. Press, 1997.

[58] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Mathematical Statistics and Probability*, pp 281–297, 1967.

[59] J. K. Martin and D. S. Hirschberg, "The time complexity of decision tree induction," Dept. Inf. Comput. Sci., Univ. California IrvineTech. Rep. 95-27 (ICS/TR-95-27), 1995.

[60] T. Mitchell, "Decision tree learning," in *Machine Learning,* New York: McGraw-Hill, 1997, pp. 52–78.

[61] B. Mobasher and R. Cooley, "Automatic personalization based on Web usage mining," *Commun. ACM*, vol. 43, no. 8, pp. 142–151, Aug. 2000.

[62] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective personalization based on association rule discovery from Web usage data," presented at the 3rd ACM Workshop Web Information and Data Management, 2001.

[63] R. Morales and H. Pain, "Modeling of novices' control skills with machine learning," in *User Modeling: Proc. 7th Int. Conf.,* UM99, pp. 159–168.

[64] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos, "Effective prediction of Web-user accesses: A data mining approach," presented at the WEBKDD 2001 Workshop, San Francisco, CA.

[65] O. Nasraoui, H. Frigui, A. Joshi, and R. Krishnapuram, "Mining Web access logs using relational competitive fuzzy clustering," presented at the 8th Int. Fuzzy Systems Association World Congr., IFSA 99, Taipei, Taiwan, R.O.C, 2006.
[66] G. Paliouras, V. Karkaletsis, C. Papathedorou, and C. Spyropoulos, "Exploiting learning techniques for the acquisition of user stereotypes and communities," presented at the Int. Conf. User Modeling, UM'99, Banff, Canada.
[67] G. Paliouras, C. Papatheodorou, V. Karkaletsis, and C. Spyropoulos, "Clustering the users of large Web sites into communities," in *Proc. 17th Int. Conf. Machine Learning*, 2000, pp. 719–726.
[68] M. Perkowitz and O. Etzioni, "Adaptive Web sites," *Commun. ACM*, vol. 43, no. 8, pp. 152–158, 2000.
[69] ——, "Adaptive Web sites: An AI challenge," in *Proc. IJCAI'98*, pp. 16–23.
[70] ——, "Towards adaptive Web sites: Conceptual framework and case study," presented at the 8th Int. World Wide Web Conf., Toronto, ON, Ontario, Canada, 1999.
[71] J. R. Quinlan, "Introduction to decision trees," in *Machine Learning*, vol. 1, 1986, pp. 81–106.
[72] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[73] L. M. Quiroga and J. Mostafa, "Empirical evaluation of explicit versus implicit acquisition of user profiles," in *Proc. 4th ACM Conf. Digital Libraries*, 1999, pp. 238–239.
[74] M. Ramsey, H. Chen, and B. Zhu, "A collection of visual thesauri for browsing large collections of geographic images," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 50, no. 9, pp. 826–834, 1999.
[75] D. Riecken, "Personalized views of personalization," *Commun. ACM*, vol. 43, no. 8, pp. 27–28, 2000.
[76] C. Romero, S. Ventura, P. de Bra, and C. de Casto, "Discovering prediction rules in AHA! courses," presented at the 9th Int. Conf. User Modeling, Lecture Notes in Artificial Intelligence, vol. 2702, Berlin, Germany: Springer-Verlag, 2003.
[77] J. Ruvini, "Adapting to the user's Internet search strategy," presented at the 9th Int. Conf. User Modeling, Lecture Notes in Artificial Intelligence, vol. 2702, Berlin, Germany: Springer-Verlag, 2003.
[78] R. R. Sarukkai, "Link prediction and path analysis using Markov chains," *Comput. Netw.*, vol. 33, no. 1–6, pp. 377–386, 2000.
[79] C. Sas, R. Reilly, and G. O'Hare, "A connectionist model of spatial knowledge acquisition in a virtual environment," presented at the 2nd Workshop Machine Learning, Information Retrieval, and User Modeling, Pittsburgh, PA, 2003.
[80] R. Schafer and T. Weyrath, "Assessing temporally variable user properties with dynamic Bayesian networks," presented at the Int. Conf. User Modeling, UM97, Sardinia, Italy, 1997.
[81] I. Schwab and W. Pohl, "Learning information interest from positive examples," in *Proc. Int. Conf. Machine Learning and Applications*, pp. 15–20, 1999.
[82] G. Semeraro, S. Ferilli, N. Fanizzi, and F. Abbattista, "Learning interaction models in a digital library service," presented at the 8th Int. Conf. User Modeling, Lecture Notes in Artificial Intelligence, vol. 2109, Berlin, Germany: Springer-Verlag, 2001.
[83] H. S. Shah, N. R. Joshi, and P. R. Wurman, "Mining for bidding strategies on eBay," in *Proc. WEBKDD 2002, 4th WEBKDD Web Mining Usage Patterns and User Profiles*, pp. 16–30.
[84] A. Sheperd, C. Watters, and A. T. Marath, "Adaptive user modeling for filtering electronic news," presented at the 35th Annu. Hawaii Int. Conf. System Sciences, HICSS-02, vol. 4.
[85] B. Shih and W. Lee, "The application of nearest neighbour algorithm on creating an adaptive on-line learning system," presented at the 31st ASEE/IEEE Frontiers in Education Conf., Reno, NV, 2001.
[86] V. Tsiriga and M. Virvou, "Initializing the student model using stereotypes and machine learning," presented at the 2002 IEEE Int. Conf. Systems, Man, and Cybernetics, Hammamet, Tunisia, 2002.
[87] M. Tsukada, T. Washio, and H. Motoda, "Automatic Web-page classification by using machine learning methods," in *Proc. Web Intelligence: Research and Development:* Lecture Notes in Artificial Intelligence, vol. 2198, Berlin, Germany: Springer-Verlag, 2001, pp. 303–313.
[88] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[89] A. Webb, *Statistical Pattern Recognition*. London, U.K.: Arnold, 1999.
[90] G. I. Webb, B. C. Chiu, and M. Kuzmycz, "Comparative evaluation of alternative induction engines for feature based modeling," *Int. J. Artif. Intell. Educ.*, vol. 8, pp. 97–115, 1997.
[91] G. I. Webb, M. J. Pazzani, and D. Billsus, "Machine learning for user modeling," in *Proc. User Modeling and User-Adapted Interaction*, 2001, vol. 11, pp. 19–29.
[92] P. Winston, "Learning by building identification trees," *Artif. Intell.*, pp. 423–442, 1992.
[93] I. H. Witten and E. Frank, *Data Mining. Practical Machine Learning Tools and Techniques With JAVA Implementations*. San Mateo, CA: Morgan Kaufman, 1999.
[94] F. Witting, "ML4UM for Bayesian network user model acquisition," presented at the 2nd Workshop Machine Learning, Information Retrieval, and User Modeling, Pittsburgh, PA, 2003.
[95] X. Zhang, "Discriminant analysis as a machine learning method for revision of user stereotypes of information retrieval systems," presented at the MLIRUM'03: 2nd Workshop Machine Learning, Information Retrieval, and User Modeling, 9th Int. Conf. User Modeling, Pittsburgh, PA, 2003.
[96] T. Zhu, R. Greiner, and G. Haubl, "Learning a model of a Web user's interests," presented at the 9th Int. Conf. User Modeling. Lecture Notes in Artificial Intelligence, vol. 2702, Berlin, Germany: Springer-Verlag, 2003.
[97] I. Zukerman and D. W. Albrecht, "Predictive statistical models for user modeling," in *Proc. User Modeling and User-Adapted Interaction*, 2001, vol. 1, pp. 5–18.
[98] I. Zukerman, D. W. Albrecht, and A. E. Nicholson, "Predicting users request on the WWW," presented at the 7th Int. Conf. User Modeling and (UM99), Banff, AB, Canada.

Enrique Frias-Martinez received the Ph.D. degree from the Polytechnic University of Madrid, Madrid, Spain, in 2000.

Currently, he is a Research Fellow in the School of Information Systems, Computing, and Mathematics, Brunel University, Uxbridge, U.K. His current research interests include soft computing, data mining, machine learning, and human–computer interaction.

Dr. Frias-Martinez was the recipient of the Best Ph.D. Thesis Award of the School of Computer Science, Polytechnic University of Madrid.

Sherry Y. Chen received the Ph.D. degree from the University of Sheffield, Sheffield, U.K., in 2000.

Currently, she is a Senior Lecturer in the School of Information Systems, Computing, and Mathematics, Brunel University, Uxbridge, U.K. She has published widely in areas such as human–computer interaction, data mining, digital libraries, and educational technology. She is the coeditor of the books, *Adaptive and Adaptable Hypermedia Systems* (IRM, 2005) and *Advances in Web-Based Education: Personalized Learning Environments* (Information Science, 2006). Her current research interests include human–computer interaction, data mining, digital libraries, and educational technology. She is a member of the editorial board of five computing journals.

Dr. Shen has given numerous invited talks, including at the 9th International Conference on User Modeling and the Engineering and Physical Sciences Research Council Network of Women in Computer Science Colloquium.

Xiaohui Liu is currently a Professor of computing at Brunel University, Uxbridge, U.K., since 2000. He is also the Honorary Pascal Professor at Leiden University, Leiden, The Netherlands, since 2004. He heads the Intelligent Data Analysis (IDA) Group. He has over 100 refereed publications in bioinformatics, data mining, intelligent systems, and image and signal processing. His current research interests include interdisciplinary research involving artificial intelligence, dynamic systems, signal processing, and statistics. He serves on the editorial boards of four computing journals, founded the Biennial International Conference Series on IDA, in 1995.

Prof. Liu is a Charted Engineer, Life Member of the American Association for Artificial Intelligence, Fellow of the Royal Statistical Society, and Fellow of the British Computer Society. He has given numerous invited talks, including a keynote at the International Conference of the Royal Statistical Society, in 2002. He serves on the editorial boards of four computing journals, founded the Biennial International Conference Series on IDA, in 1995.